LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-214068

A Simple Arbitrary Solid Slicer

Jin Yao

July 28, 2005

Journal of Computational Physics

A Simple Arbitrary Solid Slicer

Jin Yao

B-Division, Lawrence Livermore National Laboratory
Livermore, California 94551

Abstract

The intersection of a given plane and an arbitrary (possibly non-convex, with multiple connectivities) meshed solid is exactly expressed by a set of planar cross-sections. A rule for marching on the edges of an arbitrary polyhedron is set for obtaining the topology of the cross-section. The method neither seeks triangulation of the surface mesh nor utilizes look-up tables, therefore it has optimal efficiency.

INTRODUCTION

Existing algorithms for slicing an arbitrarily solid usually depend on triangulation of the surface of the solid [1]. Sometimes look-up tables may be required to determine the topology of a slice [2]. The proposed method does not take advantage of convexity, therefore triangulation is not necessary. The exact solution of the cross-section, which is in general a set of polygon sub-cross-sections, will have a reduced number of edges and less data storage with this method. The treatment is uniform for arbitrary polyhedrons, no look-up tables are needed.

DATA STRUCTURE FOR AN ARBITRARY POLYHEDRON

The smallest data set needed to specify an arbitrary polyhedron (in order to model a solid) consists of

1). *a vertex list*: an array of length N (the number of surface vertices) that contains the position of each vertex.

2). *a face list*: an array of length M (the number of faces) that contains the ID number of the vertices on each face. The ordering of the face-vertices

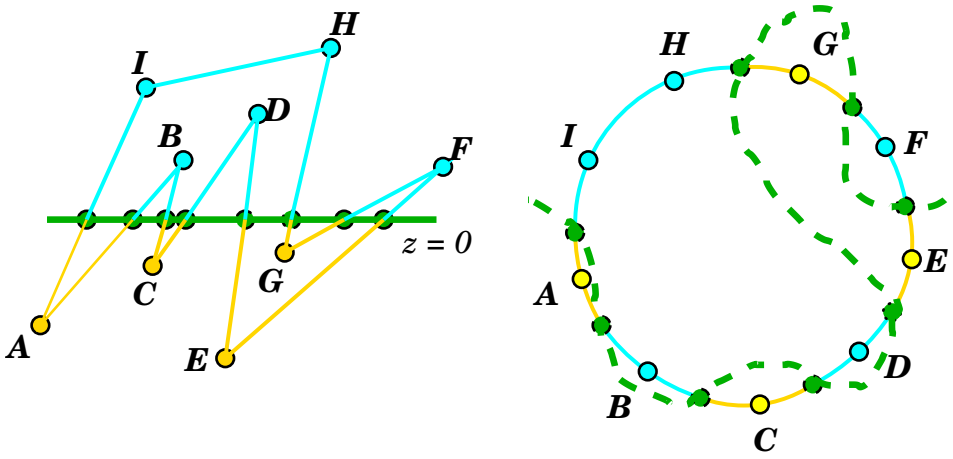

Fig. 1. *A face sliced by the $z = 0$ plane (left) and its topology (right). The blue and yellow dots are nodes. Green dots are MCPs*

is specified so that using a right hand rule, the face normal points toward outside. The neighbor faces that share edges with the current face are also specified (to define edges).

Flatness of a face **F** is not assumed, thus the intersection of a plane and **F** may or may not be a straight line-section (or possibly straight line-sections when **F** is concave). The single assumption that makes this method work is that all face-edges are straight line sections, therefore only one mesh crossing point (MCP) may exist on an edge that intersects the given plane. If **F** is not flat (for example: a bi-linear face of a twisted hexahedron), exact solution of the mesh crossing points can still be obtained. We shall calculate the MCPs and order them properly to define the topology of each sub-cross-section. In the case that a face is not flat, connecting the MCPs using straight line-sections shall give an approximate solution for an edge of the cross-section, and this solution is consistent with the correct topology. The exact solution is obtained when a face is flat.

Coloring the Surface Nodes/Edges

Without loss of generality, we assume the slicing plane is $z = 0$ in three dimensions. We 'paint' all the nodes that have non-negative z-coordinates *blue* and all other nodes *yellow*. If an edge has two blue nodes, paint it blue, if the edge has two yellow nodes, paint it yellow. If an edge has a blue node and a yellow node, paint it blue for the section above $z = 0$, yellow for the section below. The geometry of the cross-section only involves the faces that contain nodes/edges of different colors (figure 1). Therefore we make a list of these faces and discard all other faces to reduce the storage. In coding, the *color* is represented by the sign of the z-coordinates of a node.

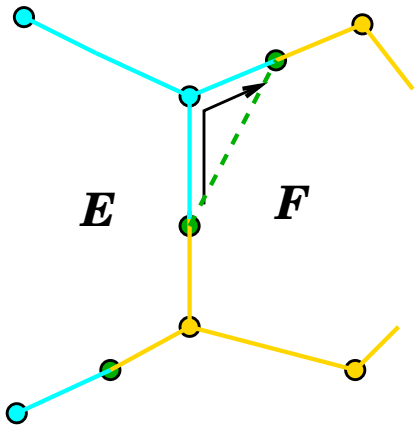

Fig. 2. *The Traffic Rule: The path of the spider on a face on the right (***F***). Dashed green line is the trace of its web.*

THE CLIMBING SPIDER ANALOGUE

We imagine a spider climbs on the surface mesh. The spider shall tighten its web at any MCP it passes by. A rule of climbing is followed so that the trace of its web would express the geometry of the boundary.

To obtain the topology, we need to determine only the order of the MCPs that the spider passes. Let us consider that the spider starts climbing (and releasing its web) at an MCP on a blue/yellow edge. One of the two faces that share this edge shall be on the spider's right side when it climbs toward the blue section on this edge. Let this face (face **F** in figure 2) be the starting face and the MCP the starting point.

The only topological information available to the spider on a given face **F** is the colors on the boundary that define **F**. To look for the next MCP, it climbs on **F**'s boundary anti-clock-wise, thus **F** is on its right side.

Staying Alive

Let us assume that the yellow paint is lethal to the spider, but blue paint is safe. Therefore, the rule of climbing on a *face* for the spider is simply:

Keep climbing on the blue part of edges until arrives at a MCP.

At this MCP, the spider would tighten its web, then start climbing on face **G** that shares this MCP with **F** (Fig. 3). Equivalently, the spider's web is a line section between the n^{th} MCP and the $(n+1)^{th}$ MCP on the boundary of a sub-cross-section of the slice.

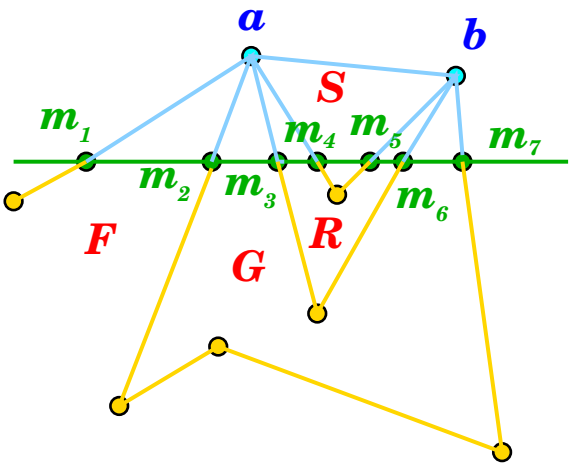

Fig. 3. *The trace of the web never returns back to a previously visited MCPs before closed. Blue dots are nodes, green dots are MCPs. Red Letters mark the faces in sight. This figure is a topology map, not necessarily reflecting the exact geometry.*

One may consider a face as a *'city block'*, a face-edge a *'street'*, a node an *'intersection of streets'*, and an MCP as a *'road block'* in the middle of a street. Then the above *rule of climbing* is equivalent to:

> *At an intersection (a* **node***), turn into the right most street (a* **face-edge***). If a road block (a* **MCP***) is arrived at, make a U-turn. Keep going until the first road block (the starting* **MCP***) is arrived at again.*

The order in which the MCPs are visited determines the topology. When the starting MCP is visited again by the spider, the trace of the spider's web is closed. At this moment one shall remove the spider. If the surface of the given solid is non-convex, holes, and sub-cross sections with multiple boundaries may exist. In this situation, there may exist MCPs that has not been visited by the spider after one climbing, we put the spider on such an MCP and let the spider start climbing again, until all the MCPs have been visited.

Never Going Back

It is natural to assume that a node is shared by at least *three* faces. Because the spider always turns into the right most edge at a node by following the traffic rule, it shall never return to a previously visited MCP, until the trace of its web is closed. As Fig. 3 shows (the spider starts climbing from $\mathbf{m_1}$), although the spider would make a U-turn at the MCP ($\mathbf{m_2}$) shared by **F** and **G**, it would surely arrive at a blue node (**a**) shared by these two faces. The spider would find an edge on **G** that is defined by blue node **a** and either a blue node, or a yellow node. In either case this edge is at least partially painted blue starting from **a**. Thus, the spider would make a right turn on the boundary of **G** at node **a**, and not go back to $\mathbf{m_1}$.

In figure 3, the spider's path on **F** is $\mathbf{m_1} \rightarrow \mathbf{a} \rightarrow \mathbf{m_2}$. Then it arrives at face **G**, and climbs along $\mathbf{m_2} \rightarrow \mathbf{a} \rightarrow \mathbf{m_3}$ before arriving face **R**. The spider's path on **R** is $\mathbf{m_3} \rightarrow \mathbf{a} \rightarrow \mathbf{m_4}$. Then it arrives at face **S**, continue climbing through $\mathbf{m_4} \rightarrow \mathbf{a} \rightarrow \mathbf{b} \rightarrow \mathbf{m_5}$. At this moment the spider returns to face **R** and its second path on **R** is $\mathbf{m_5} \rightarrow \mathbf{b} \rightarrow \mathbf{m_6}$. Then it goes back to face **G** again, however continue its climb along $\mathbf{m_6} \rightarrow \mathbf{b} \rightarrow \mathbf{m_7}$. It is clear that the spider shall keep turning right at a node and never return to a previously visited MCP, until the starting MCP is found again, and the web released by the spider is closed.

The 'climbing' on a face is mathematically performed by looping over the vertices of the given face, and by checking the nodal **z**-coordinate. The location of a MCP can obtained by linear interpolation of nodal function values.

Connectivity of the Enclosed Region

The 'spider' has never left blue sections while climbing, therefore the region enclosed by the web is a *connected* one by the spider's continuous climbing.

The *rule of climbing* ensures that a blue section of face-boundary is always on the left side of the trace of the web, and yellow sections on the right. Therefore the trace of the web defines a boundary between blue and yellow sections. Then we have obtained the correct topology. Because every MCP is on the $z = 0$ plane, the 'spider's web' (paths of straight line-sections) is naturally lying on the plane $z = 0$, and the correct geometry is also obtained.

Procedure of Implementation

The cost of the proposed algorithm is linear. The slicing plane $z = 0$ can be replaced by any continuous surface $f(x, y, z) = 0$. The z-coordinate at a node i can be replaced by nodal function value $f(x_i, y_i, z_i)$. The implementation of the proposed method has two steps and is explained as the following

Preparation:

1). Calculate the location of each MCP (the intersection of $f(x, y, z) = 0$ and an edge), for each cell-edge that contains an MCP.
2). Set $m = 0$, and $n = 0$. All edges are assumed unmarked.

Searching:

1). Loop over all the edges that contains an MCP, find an unmarked edge.
2). Mark this edge, and label the MCP on this edge $[m, n]$. Then look for the node on this edge for which $f > 0$, find the face **F** on the right.
3). Starting from the current edge, one 'marches' on the edges of face **F** anti-clockwise, until he meets another edge that contains an MCP.

4). If this new edge is unmarked, increase n by 1, go back to step 2). If this new edge is marked, then sub-cross-section m (which has n vertices) is obtained. Record this sub-cross section, then increase m by 1 and reset n to 0.
5). Go to 1) and continue the loop, until all MCPs are labeled.

The General Case

The only topological requirement for the 'climbing spider' analog to work is that there is at most *one* MCP on an edge. Therefore the analog can be used to find the topology of an arbitrary continuous front (not necessarily a plane) intersecting an arbitrarily meshed closed surface. In this case, occasionally an edge will contain multiple MCPs if the front-curvature is locally high, thus the 'spider' may be trapped in a blue section between two adjacent MCPs on an edge intersected multiple times by the front.

This situation creates almost no complications. One may consider it as a degenerat case of a sub-cross-section that has only two edges and zero area (equivalent to discarding the two MCPs). The topology of the solution is not well defined on a multiply intersected edge. However it will not cause unacceptable numerical errors if the local resolution of the mesh is adequate for defining the curvature of the front shape. Furthermore, the error caused by neglecting degenerat sub-cross-sections can be reduced by local curve fitting for the boundary.

APPLICATION

This method can be used to slice complex objects such as regions with holes and concave boundaries in image processing practice. However, our main interest is to define cross-sections of interfaces contained in arbitrarily meshed regions. Iso-surface capturing using nodal function values, and interface reconstruction using volume of fluids methods are our primary interest.

Iso-surface Capturing

The purpose is to define an iso-surface $f(\vec{r}, t) = 0$ given nodal function values for an arbitrary mesh that may evolve with time. The face of a cell is assumed not triangulated (for example, the case of a hexahedral mesh). The iso-surface (in general not a plane) is the collection of cross-sections contained in cells with alternate signs of nodal function values. For such a cell, the MCPs can be obtained by local surface fitting using a plane. The proposed method can correctly connect the MCPs to form the boundary of the cross-section (or a set

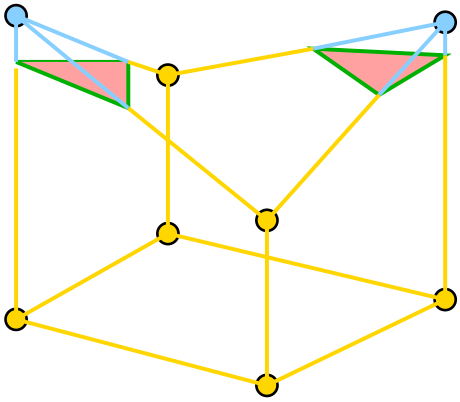

Fig. 4. *A hexahedron is sliced by a plane. Multiple sub-cross-sections are produced.*

of sub-cross-sections for a highly distorted cell). Look-up tables such as used by a marching-cubes method are not required, because the proposed method automatically finds all the sub-pieces. In the special case that $f(\vec{r}, t)$ is the signed distance, the proposed method has been used to construct a simple narrow-band method to propagate interfaces on three-dimensional arbitrary meshes [3].

Interface Reconstruction

Interface reconstruction for an arbitrary mesh requires one to match a given partial cell-volume by shifting a plane of specified slope in a cell. The cross-section area is used in a Newton-Raphson method for determine the derivative of the partial volume in the normal direction. One first sets the plane somewhere in the cell (at the cell center, say). Let s be the distance between the plane $\mathbf{P}(s)$ to the cell-center, $V(s)$ be the partial cell volume below the plane and V^* be the volume to match, and $A(s)$ be the area of the cross-section sliced by a plane $\mathbf{P}(s)$, then a second order scheme for partial volume matching is

$$s_{n+1} = s_n - \frac{V(s_n) - V^*}{A(s_n)}. \tag{1}$$

This scheme usually converges to 10^{-6} accuracy with three or four iterations.

The area calculation requires only linear combinations of products of coordinates after the order of the vertices is obtained. In the case that the cross-section is defined by multiple boundaries, the area of holes is automatically deducted from the sum of the areas because the area enclosed is signed by the right hand rule, which is consistent with the *rule of climbing*.

CONCLUSION

A simple method for determining the topology of the cross-section of a closed arbitrary surface meshed with arbitrary polygon faces, sliced by a plane, is proposed.

The method can treat the general case in a uniform way, thus look-up tables are not needed for determining topology. No triangulation is required for the method to work either because convexity is not needed. As a result, data-structures can be simplified and the storage reduced. The method has been applied for iso-surface capturing and interface-reconstruction with a volume of fluids method.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. The author wishes to thank Richard Sharp of Lawrence Livermore National Laboratory for his guidance.

References

[1] Rock, Stephen J. and Wozny, Michael J., Utilizing Topological Information to Increase Scan Vector Generation Efficiency, Solid Free-form Fabrication Symposium Proceedings, pp. 28-36, (1991).

[2] Lorensen, William E., and Cline Harvey E., Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, Vol. 21, 4, pp. 163-169,(1987)

[3] Yao, J., A Fast Three-Dimensional Lighting Time Algorithm, Proceedings of the Conference of the American Physical Society Topical Group on Shock Compression of Condensed Matter, Portland, Oregon, pp. 421-424, (2003).

[4] Zwillinger D., Standard Mathematical Tables and Formulae, the 30th edition, CRC Press, (1996).